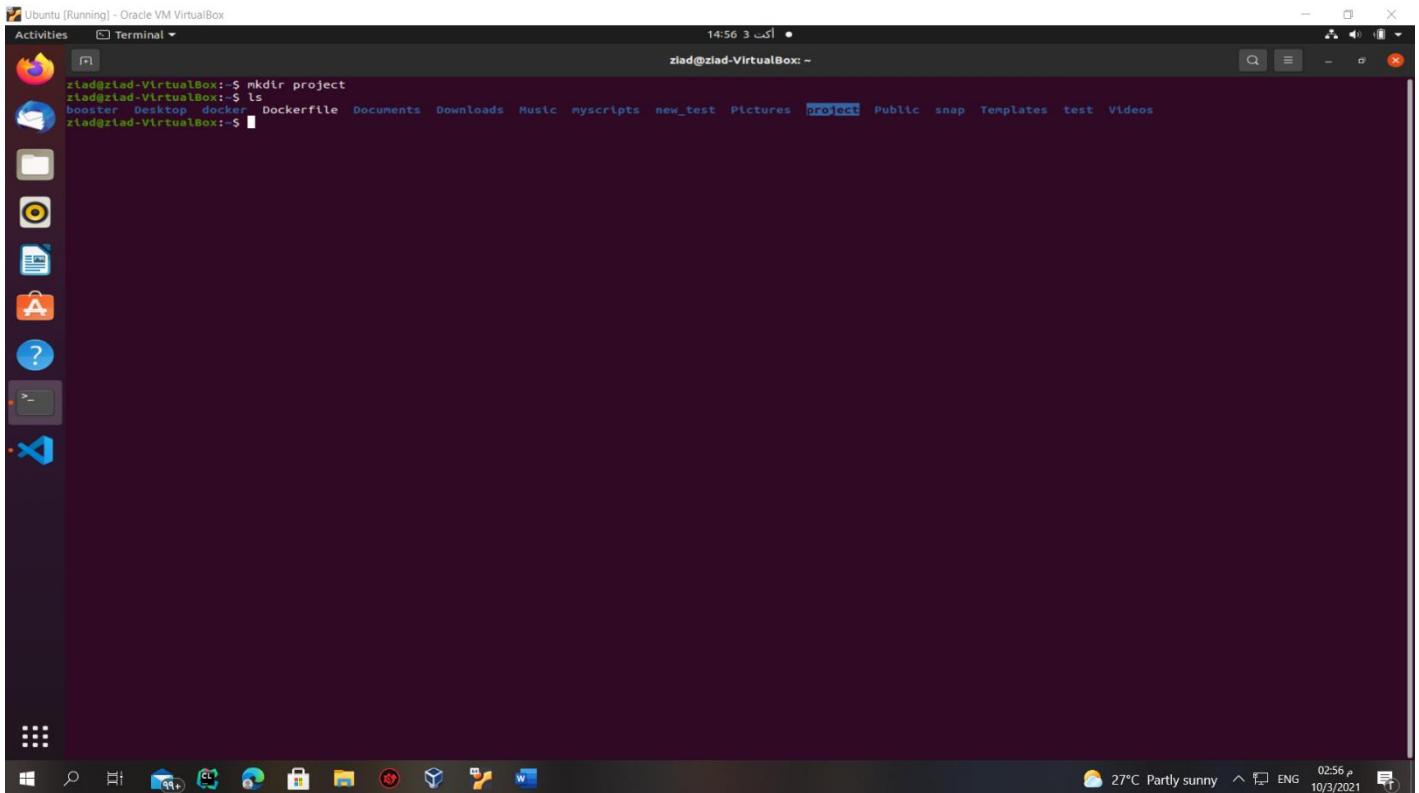
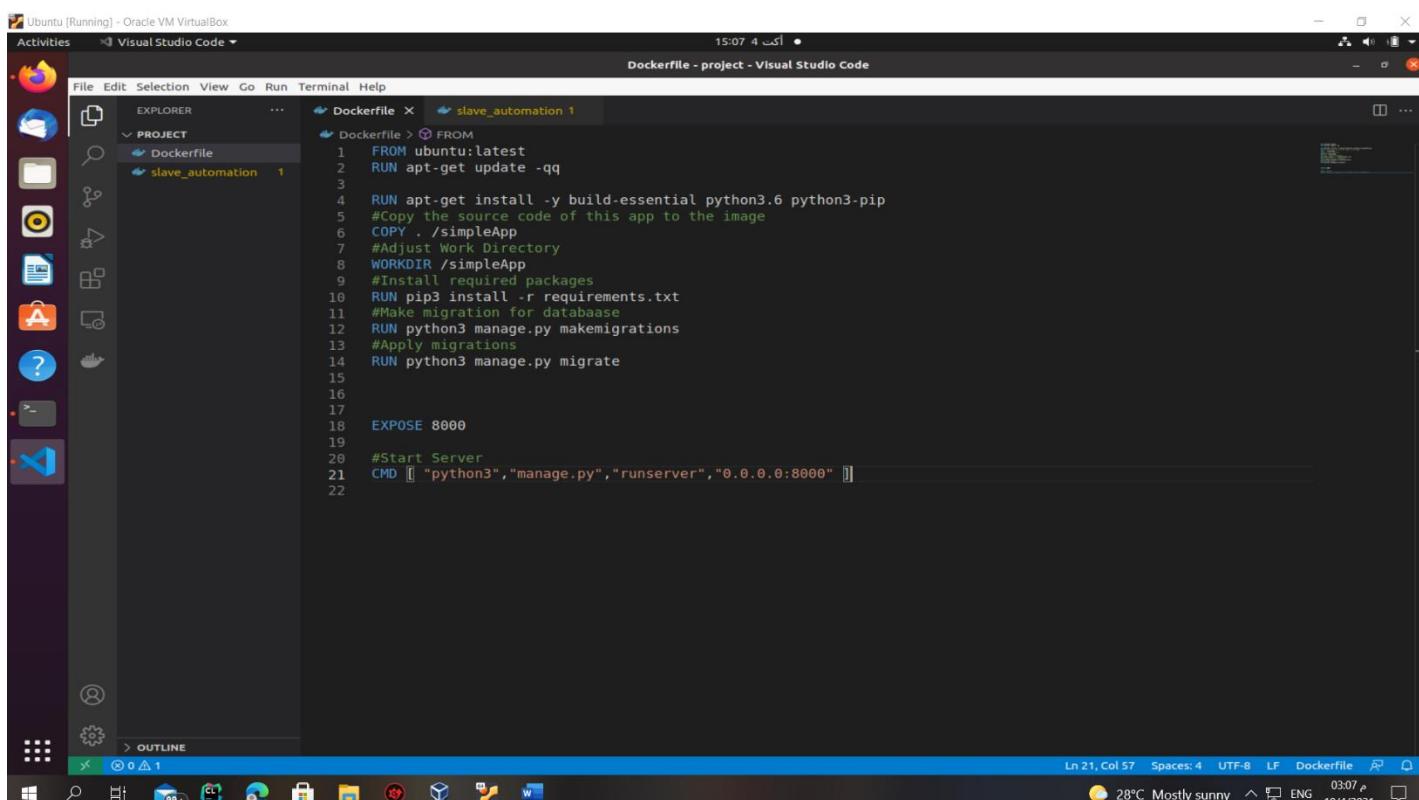


Deploying a Django App: The Process

1-Create A project directory

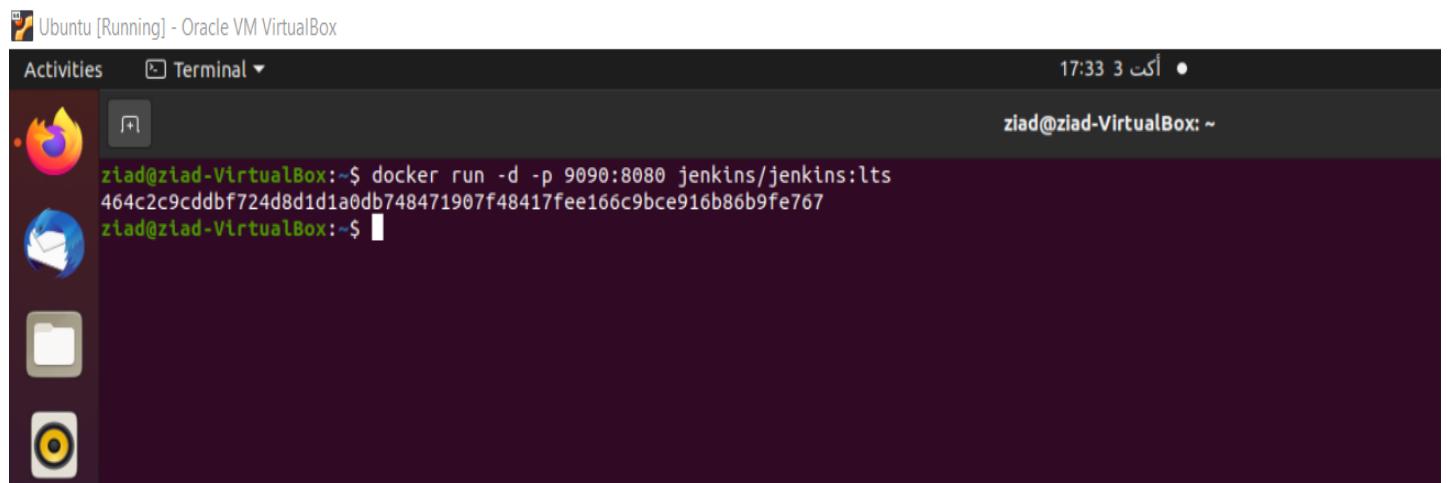


2- Create Docker File for the Django App inside Directory



4-Now let's start a Jenkins Server

- a) We will start Jenkins as a docker container on port 9090



A screenshot of an Ubuntu desktop environment within Oracle VM VirtualBox. The desktop has a dark theme with icons for the Dash, Activities, Terminal, and Home. A terminal window is open at the bottom right, showing the command:

```
ziad@ziad-VirtualBox:~$ docker run -d -p 9090:8080 jenkins/jenkins:lts  
464c2c9cddbf724d8d1d1a0db748471907f48417fee166c9bce916b86b9fe767  
ziad@ziad-VirtualBox:~$
```

- b) Now we configure the Jenkins server

i) Install Role-Based Authorization Strategy

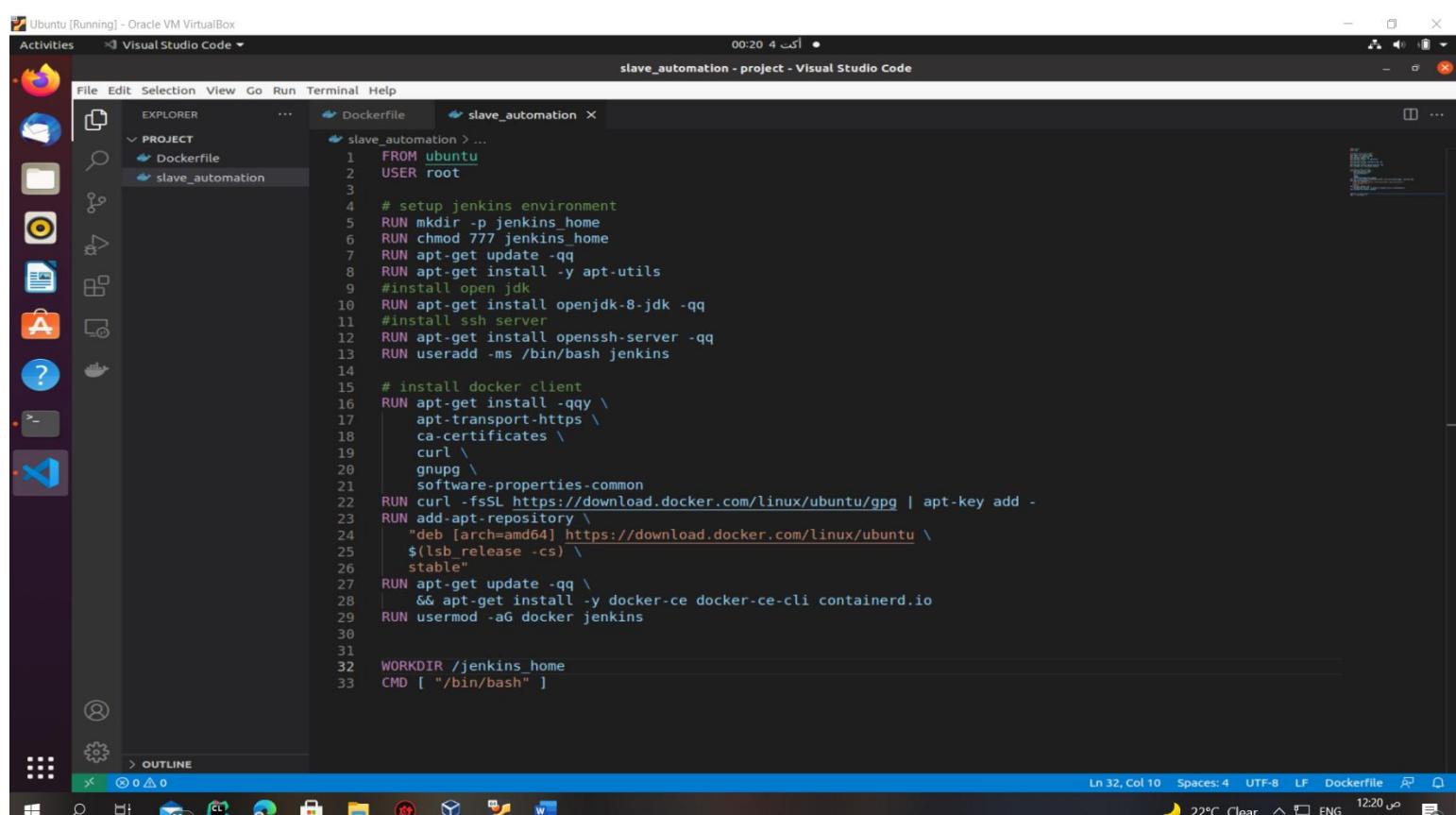
The screenshot shows the Jenkins Plugin Manager interface. The left sidebar includes links for Dashboard, Manage Jenkins, and Update Center. The main area has a search bar at the top. Below it, tabs for 'Updates', 'Available' (which is selected), 'Installed', and 'Advanced' are shown. A table lists available plugins. The first plugin listed is 'Role-based Authorization Strategy' under the 'Security' category, which is currently selected. It has a brief description: 'Enables user authorization using a Role-Based strategy. Roles can be defined globally or for particular jobs or nodes selected by regular expressions.' The version is 3.2.0 and it was released 2 months and 2 days ago. Below the table are buttons for 'Install without restart', 'Download now and install after restart', and 'Check now'.

II) Create A multibranch Pipeline

The screenshot shows the Jenkins dashboard for the 'Django-App' pipeline. The left sidebar contains links for Up, Status, Configure, Scan Multibranch Pipeline Now, Scan Multibranch Pipeline Log, Multibranch Pipeline Events, Delete Multibranch Pipeline, People, Build History, Rename, Pipeline Syntax, Credentials, and New View. The main area displays the 'Django-App' pipeline configuration. It shows three branches: 'development', 'master', and 'production'. Each branch has a green checkmark icon, indicating they are healthy. The 'development' branch's last success was 36 seconds ago. The 'master' branch's last success was 36 seconds ago. The 'production' branch's last success was 37 seconds ago. The 'Last Failure' column shows 'N/A' for all branches. The 'Last Duration' column shows 26 seconds for development, 31 seconds for master, and 29 seconds for production. At the bottom, there are links for 'Atom feed for all', 'Atom feed for failures', and 'Atom feed for just latest builds'.

5- Now, what we need is to automate the process of setting up the Jenkins environment, and add the docker client in the slave to be able to use docker, rather than manually setting up the Jenkins environment.

a) Create a docker image than sets up the Jenkins environment and adds the docker client.



The screenshot shows a Visual Studio Code interface running on an Ubuntu virtual machine. The title bar indicates it's an Oracle VM VirtualBox instance. The main area displays a Dockerfile with the following content:

```
FROM ubuntu
USER root

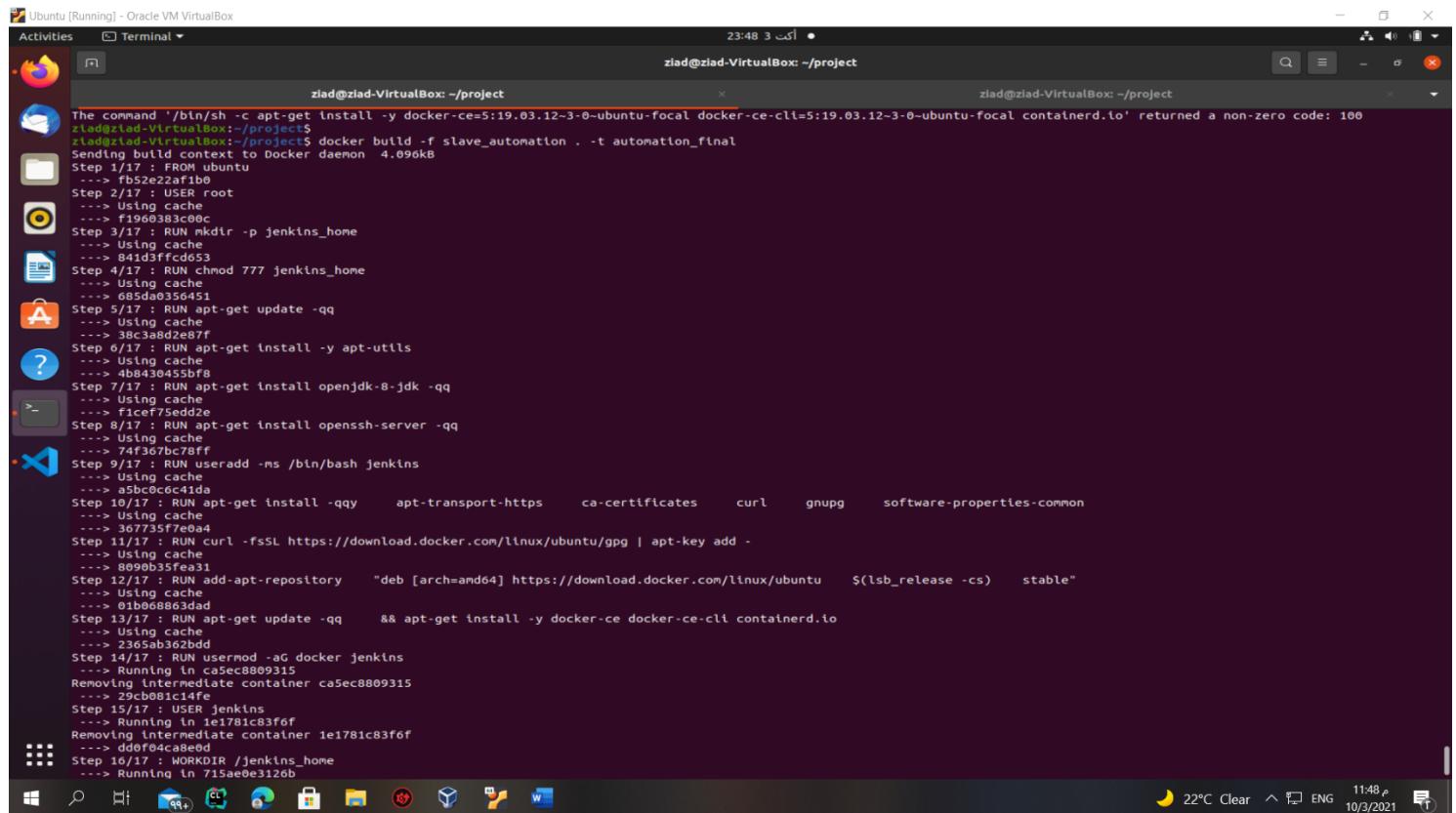
# setup jenkins environment
RUN mkdir -p jenkins_home
RUN chmod 777 jenkins_home
RUN apt-get update -qq
RUN apt-get install -y apt-utils
#install open jdk
RUN apt-get install openjdk-8-jdk -qq
#install ssh server
RUN apt-get install openssh-server -qq
RUN useradd -ms /bin/bash jenkins

# install docker client
RUN apt-get install -qqy \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg \
    software-properties-common
RUN curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -
RUN add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) \
    stable"
RUN apt-get update -qq \
    && apt-get install -y docker-ce docker-ce-cli containerd.io
RUN usermod -aG docker jenkins

WORKDIR /jenkins_home
CMD [ "/bin/bash" ]
```

The code block includes line numbers from 1 to 33. The Dockerfile starts by pulling the Ubuntu base image and switching to the root user. It then creates a directory for Jenkins and sets its permissions. It updates the package list, installs apt-utils, and installs the OpenJDK 8 Java Development Kit. It also installs the SSH server. Then, it adds a new user named 'jenkins' with a home directory at '/jenkins_home'. Finally, it installs the Docker CE package, adds the 'jenkins' user to the 'docker' group, and sets the default command to run a bash shell.

b) Build docker image



The screenshot shows a terminal window titled "Ubuntu [Running] - Oracle VM VirtualBox". The command being run is:

```
ziad@ziad-VirtualBox:~/project$ docker build -f slave_automation . -t automation_final
```

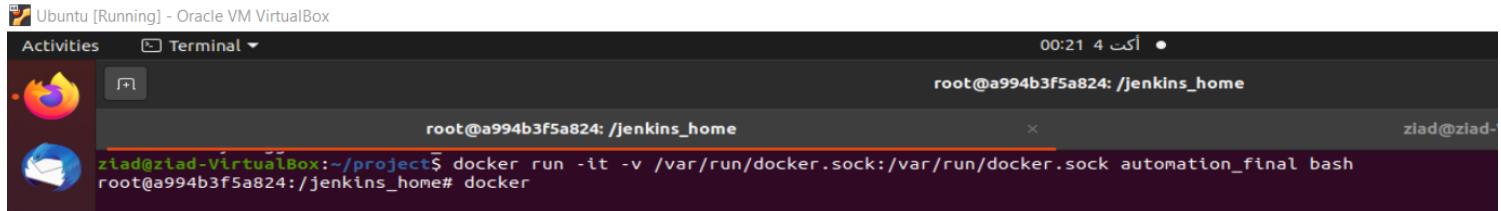
The output of the command is as follows:

```
The command '/bin/sh -c apt-get install -y docker-ce=5:19.03.12-3~ubuntu-focal docker-ce-cli=5:19.03.12-3~ubuntu-focal containerd.io' returned a non-zero code: 100
Sending build context to Docker daemon 4.096kB
Step 1/17 : FROM ubuntu
--> fb52e22af1b0
Step 2/17 : USER root
--> Using cache
---- f1960383c00c
Step 3/17 : RUN mkdir -p jenkins_home
--> Using cache
---- 841d3ffcd653
Step 4/17 : RUN chmod 777 jenkins_home
--> Using cache
---- 841d3ffcd651
Step 5/17 : RUN apt-get update -qq
--> Using cache
---- 38c3a8d2e07f
Step 6/17 : RUN apt-get install -y apt-utils
--> Using cache
---- 4b8430455bbf8
Step 7/17 : RUN apt-get install openjdk-8-jdk -qq
--> Using cache
---- f1cef75ed2ze
Step 8/17 : RUN apt-get install openssh-server -qq
--> Using cache
---- 74f367bc78ff
Step 9/17 : RUN useradd -ms /bin/bash jenkins
--> Using cache
---- a5bc0c6c41da
Step 10/17 : RUN apt-get install -qqy apt-transport-https ca-certificates curl gnupg software-properties-common
--> Using cache
---- 36775f1f9e04
Step 11/17 : RUN curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -
--> Using cache
---- 8090b35fea31
Step 12/17 : RUN add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
--> Using cache
---- 01b068863dad
Step 13/17 : RUN apt-get update -qq && apt-get install -y docker-ce docker-ce-cli containerd.io
--> Using cache
---- 2365ab362bdd
Step 14/17 : RUN usermod -aG docker jenkins
--> Running in ca5ec8809315
Removing Intermediate container ca5ec8809315
---- 29cb001c14fe
Step 15/17 : USER jenkins
--> Running in 1e1781c83f6f
Removing Intermediate container 1e1781c83f6f
---- dd0f0ea8e0d
Step 16/17 : WORKDIR /jenkins_home
--> Running in 715ae0e03126b
```

The terminal window also shows system status at the top right, including the date and time (11:48 PM, 10/3/2021), and a taskbar at the bottom.

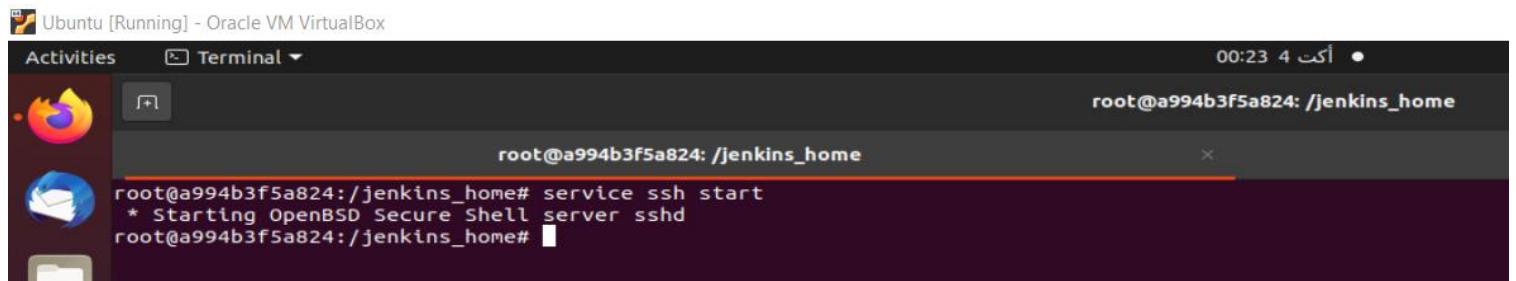
c) Now run the docker container, and bind mount the /var/run/docker.sock file to be able to use docker

outside of docker rather than docker inside of docker, which is a bad practice.



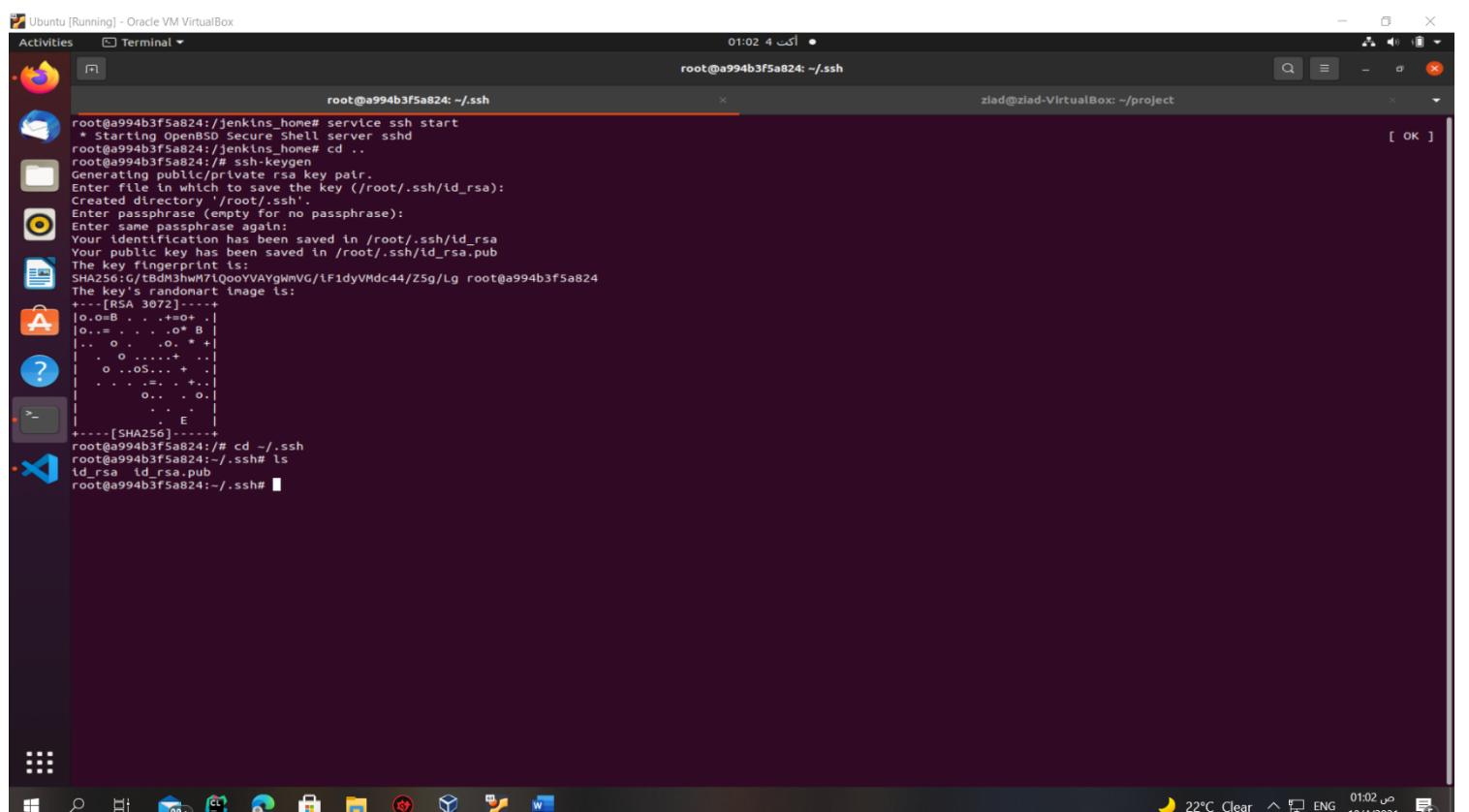
A screenshot of a Linux desktop environment showing a terminal window. The terminal title is "root@a994b3f5a824: /jenkins_home". The command entered is "docker run -it -v /var/run/docker.sock:/var/run/docker.sock automation_final bash". The terminal window is part of the Unity interface, with other icons like a browser and file manager visible in the background.

e) start ssh



A screenshot of a Linux desktop environment showing a terminal window. The terminal title is "root@a994b3f5a824: /jenkins_home". The command entered is "service ssh start". The output shows the service starting successfully. The terminal window is part of the Unity interface.

f) generate ssh-keygen, and add public key in authorized_keys file



A screenshot of a Linux desktop environment showing a terminal window. The terminal title is "root@a994b3f5a824: ~/.ssh". The command entered is "service ssh start". The output shows the service starting successfully. The terminal window is part of the Unity interface. The user then runs "ssh-keygen" to generate an RSA key pair. The command "cd .." is used to move up one directory. The command "ls" is used to list files in the current directory, showing "id_rsa" and "id_rsa.pub". The desktop environment includes icons for a browser, file manager, and terminal.

g) now create a node, and configure it, the host should be the ip address of the docker container gotten by docker inspect command

The screenshot shows the Jenkins web interface on a Linux desktop. The user is configuring a new node named 'docker'. The configuration details are as follows:

- Name:** docker
- Description:** (empty)
- Number of executors:** 1
- Remote root directory:** /root/jenkins
- Labels:** docker
- Usage:** Use this node as much as possible
- Launch method:** Launch agents via SSH
- Host:** 172.17.0.3
- Credentials:** root (selected dropdown)
- Host Key Verification Strategy:** Non verifying Verification Strategy
- Availability:** (checkboxes for various availability options)

The browser address bar shows the URL: `localhost:9090/computer/docker/configure`. The Jenkins dashboard sidebar is visible on the left.

h) Add a credential, and add inside it the public key

The screenshot shows a Firefox browser window running on an Ubuntu system within a VirtualBox environment. The URL in the address bar is `localhost:9090/credentials/store/system/domain/_/credential/ubuntu/update`. The Jenkins UI is displayed, showing the 'Global credentials (unrestricted)' screen for the 'root' user. On the left, there's a sidebar with icons for Dashboard, Credentials, System, and Global credentials. The 'Update' option under 'Credentials' is selected. The main form fields include:

- Scope:** Global (Jenkins, nodes, items, all child items, etc)
- ID:** ubuntu
- Description:** (empty)
- Username:** root
- Treat username as secret:** (unchecked)
- Private Key:** Enter directly (radio button selected). A text area labeled 'Key' contains the text 'Concealed for Confidentiality'. There is a 'Replace' button next to the text area.
- Passphrase:** (empty)

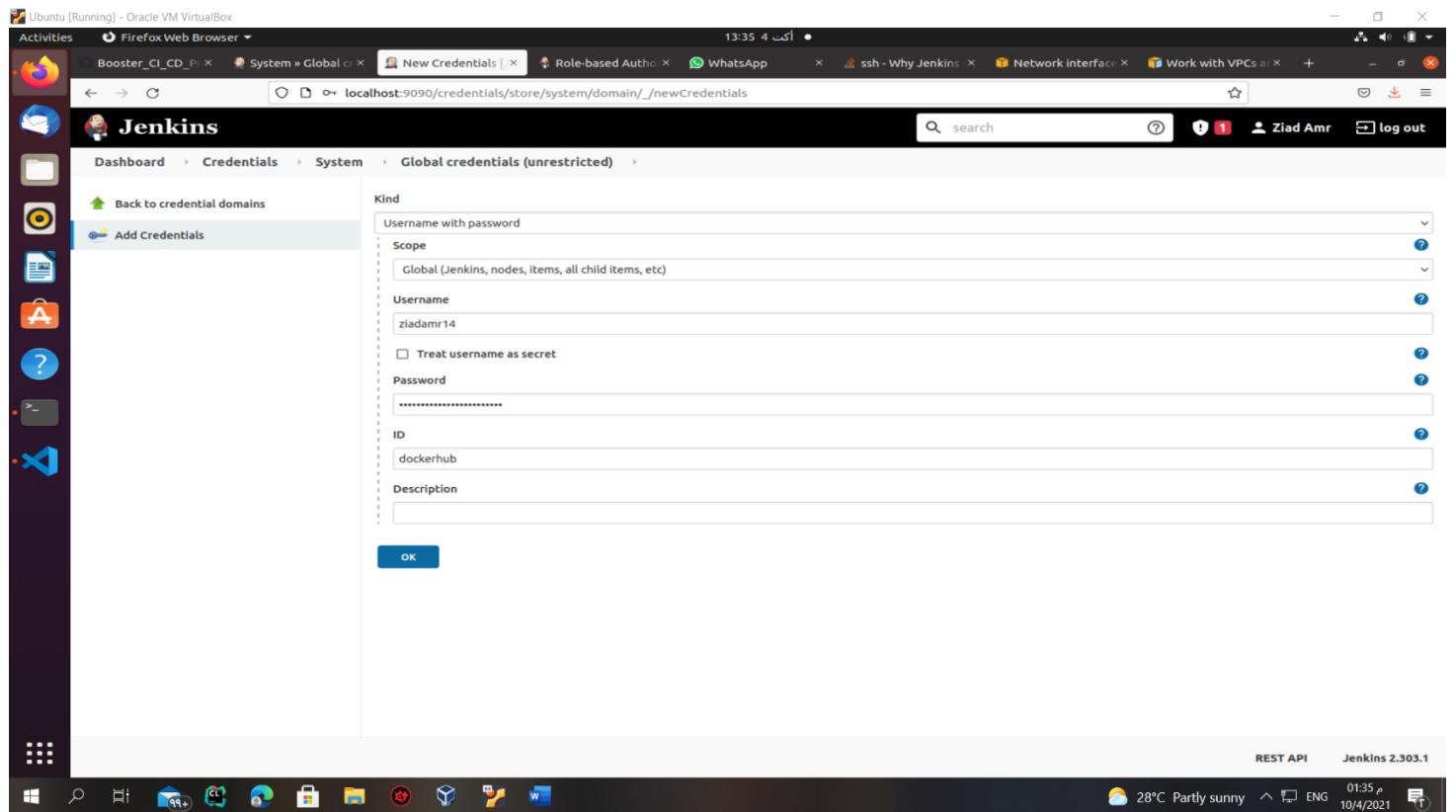
At the bottom of the form are 'Save' and 'Cancel' buttons. The status bar at the bottom right shows 'REST API' and 'Jenkins 2.303.1'.

i)Check that node is successfully authenticated

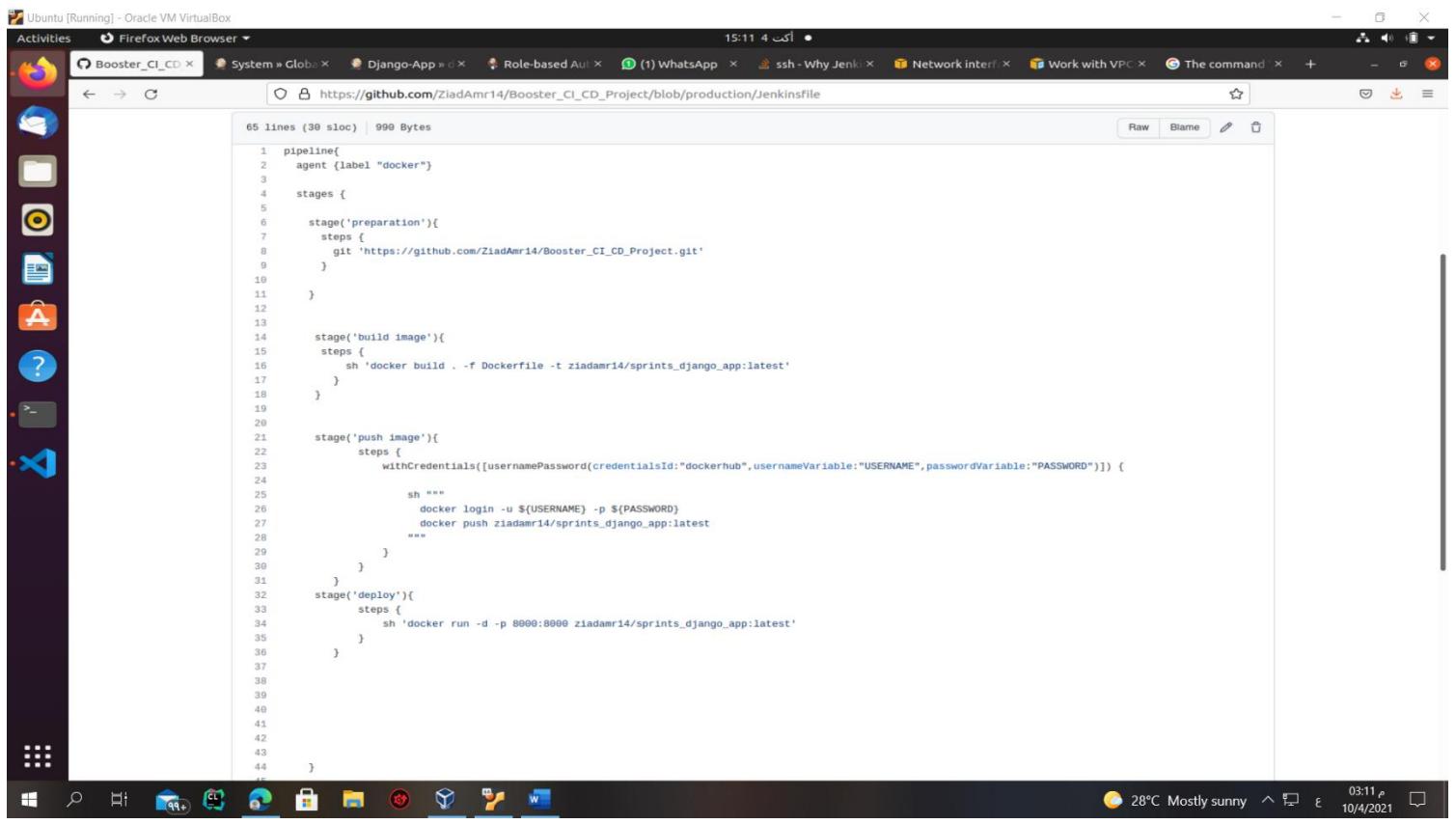
The screenshot shows a Linux desktop environment with a docked application bar. A Firefox browser window is open to the Jenkins 'Nodes' page. The browser's address bar shows 'localhost:9090/computer/'. The Jenkins interface displays two nodes: 'docker' and 'master'. Both nodes are listed as 'idle' under the 'Build Executor Status' section. The system tray at the bottom right of the screen shows the date as 10/4/2021, time as 01:12, and weather as 27°C Partly sunny.

6)Now that everything is configured, we need to write the Jenkins file, we will split the stages into preparation stage, build stage, push stage, deploy stage, and notification stage when we add the slack plugin

a) first, we need to add a credential for the docker hub username and password.



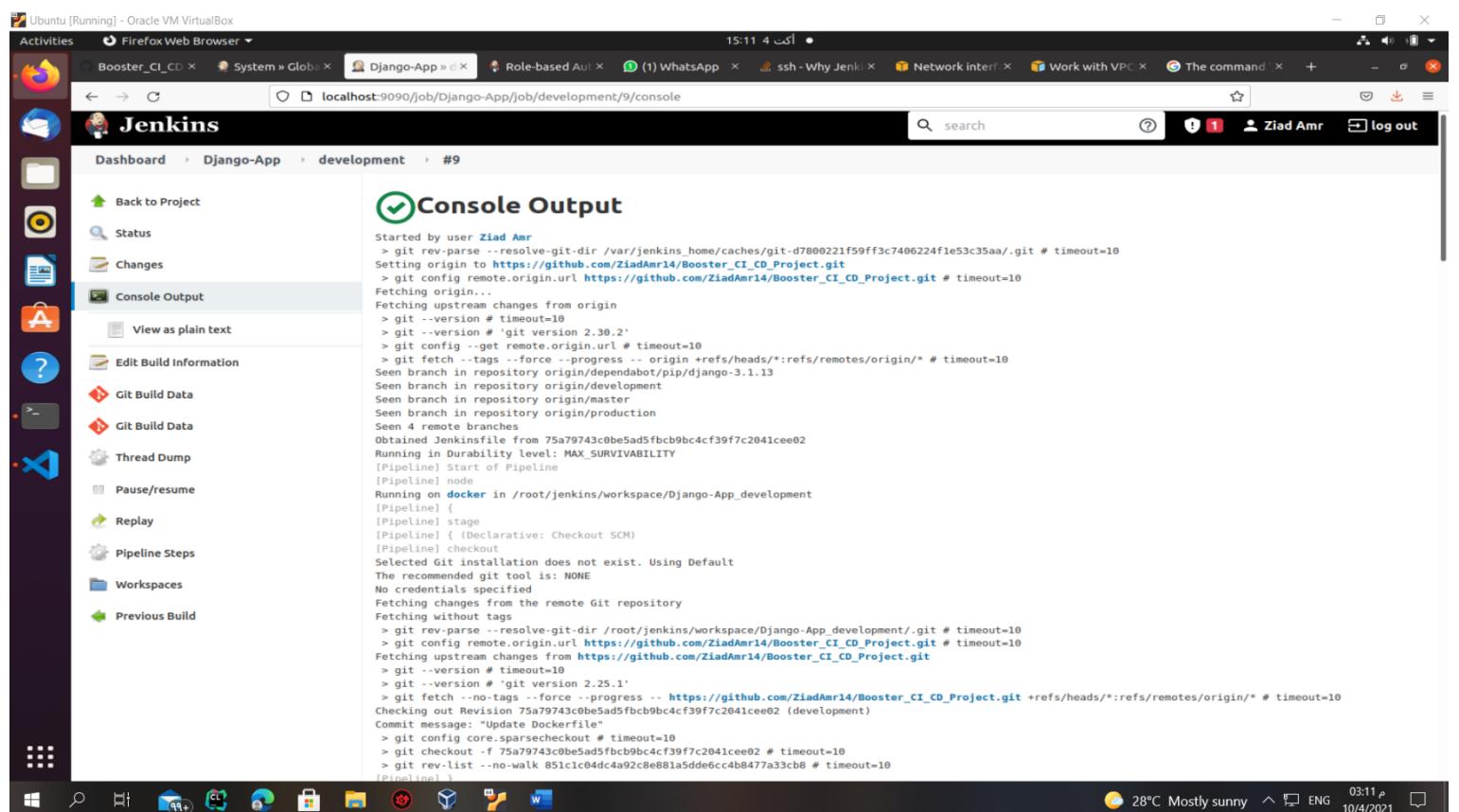
b) Now here is the Jenkins file



A screenshot of a Firefox browser window titled "Ubuntu [Running] - Oracle VM VirtualBox". The address bar shows the URL https://github.com/ZiadAmr14/Booster_CI_CD_Project/blob/production/Jenkinsfile. The page content displays a Jenkinsfile with 65 lines of code. The Jenkinsfile defines a pipeline with stages for preparation, building an image, pushing the image to Dockerhub, and deploying it. It uses Docker and Jenkins commands like git, docker build, docker login, and docker push.

```
65 lines (30 sloc) 990 Bytes
1 pipeline{
2     agent {label "docker"}
3
4     stages {
5
6         stage('preparation'){
7             steps {
8                 git 'https://github.com/ZiadAmr14/Booster_CI_CD_Project.git'
9             }
10        }
11    }
12
13    stage('build image'){
14        steps {
15            sh 'docker build . -f Dockerfile -t ziadamr14/sprints_django_app:latest'
16        }
17    }
18 }
19
20
21    stage('push image'){
22        steps {
23            withCredentials([usernamePassword(credentialsId:"dockerhub",usernameVariable:"USERNAME",passwordVariable:"PASSWORD")]) {
24
25                sh """
26                    docker login -u ${USERNAME} -p ${PASSWORD}
27                    docker push ziadamr14/sprints_django_app:latest
28                """
29            }
30        }
31    }
32    stage('deploy'){
33        steps {
34            sh 'docker run -d -p 8000:8000 ziadamr14/sprints_django_app:latest'
35        }
36    }
37
38
39
40
41
42
43
44
45 }
```

c)Build pipeline



A screenshot of a Firefox browser window titled "Ubuntu [Running] - Oracle VM VirtualBox". The address bar shows the URL <localhost:9090/job/Django-App/job/development/9/console>. The page title is "Jenkins". The left sidebar shows navigation options like Dashboard, Back to Project, Status, Changes, and Console Output. The main content area is titled "Console Output" and shows the Jenkinsfile being parsed and executed. The output includes commands like git rev-parse, git config, git fetch, and docker run, along with messages about fetching upstream changes and setting up the Docker environment.

```
Started by user Ziad Amr
> git rev-parse --resolve-git-dir /var/jenkins_home/.git # timeout=10
Setting origin to https://github.com/ZiadAmr14/Booster_CI_CD_Project.git
> git config remote.origin.url https://github.com/ZiadAmr14/Booster_CI_CD_Project.git # timeout=10
Fetching origin...
Fetching upstream changes from origin
> git --version # timeout=10
> git --version # 'git version 2.30.2'
> git config --get remote.origin.url # timeout=10
> git fetch --tags --force --progress --origin +refs/heads/*:refs/remotes/origin/* # timeout=10
Seen branch in repository origin/dependabot/pip/django-3.1.13
Seen branch in repository origin/development
Seen branch in repository origin/master
Seen branch in repository origin/production
Seen 4 remote branches
Obtained Jenkinsfile from 75a79743c0be5ad5fbcb9bc4cf39f7c2041cee02
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on docker in /root/jenkins/workspace/Django-App_development
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
Fetching changes from the remote Git repository
Fetching without tags
> git rev-parse --resolve-git-dir /root/jenkins/workspace/Django-App_development/.git # timeout=10
> git config remote.origin.url https://github.com/ZiadAmr14/Booster_CI_CD_Project.git # timeout=10
Fetching upstream changes from https://github.com/ZiadAmr14/Booster_CI_CD_Project.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
> git fetch --no-tags --force --progress -- https://github.com/ZiadAmr14/Booster_CI_CD_Project.git +refs/heads/*:refs/remotes/origin/* # timeout=10
Checking out Revision 75a79743c0be5ad5fbcb9bc4cf39f7c2041cee02 (development)
Commit message: "Update Dockerfile"
> git config core.sparsecheckout # timeout=10
> git checkout -f 851c1c4dc49a92c8e881a5dde6c4ab47a733cb8 # timeout=10
> git rev-list --no-walk 851c1c4dc49a92c8e881a5dde6c4ab47a733cb8 # timeout=10
[Pipeline]
```

d) Here is the image on dockerhub

The screenshot shows a Firefox browser window running on an Ubuntu desktop environment within Oracle VM VirtualBox. The URL in the address bar is https://hub.docker.com/repository/docker/ziadamlr14/sprints_django_app. The Docker Hub interface is visible, showing the repository details for 'ziadamlr14/sprints_django_app'. The repository has 1 tag, 'latest', which was pushed 8 minutes ago. There is a note that vulnerability scanning is disabled. The Docker commands section shows the command 'docker push ziadamlr14/sprints_django_app:tagname'. The 'Automated Builds' section indicates that manually pushing images to Hub is possible by connecting accounts from GitHub or Bitbucket.

Docker is updating and extending our product subscriptions. Please read our blog for more information.

docker hub

ziadamlr14 / sprints_django_app

General Tags Builds Collaborators Webhooks Settings

Advanced Image Management

View preview

ziadamlr14 / sprints_django_app

This repository does not have a description

Last pushed: 8 minutes ago

Docker commands

To push a new tag to this repository,

docker push ziadamlr14/sprints_django_app:tagname

Tags and Scans

VULNERABILITY SCANNING - DISABLED

Enable

TAG OS PULLED PUSHED

latest 8 minutes ago 8 minutes ago

See all

Automated Builds

Manually pushing Images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available on Pro and Team plans.

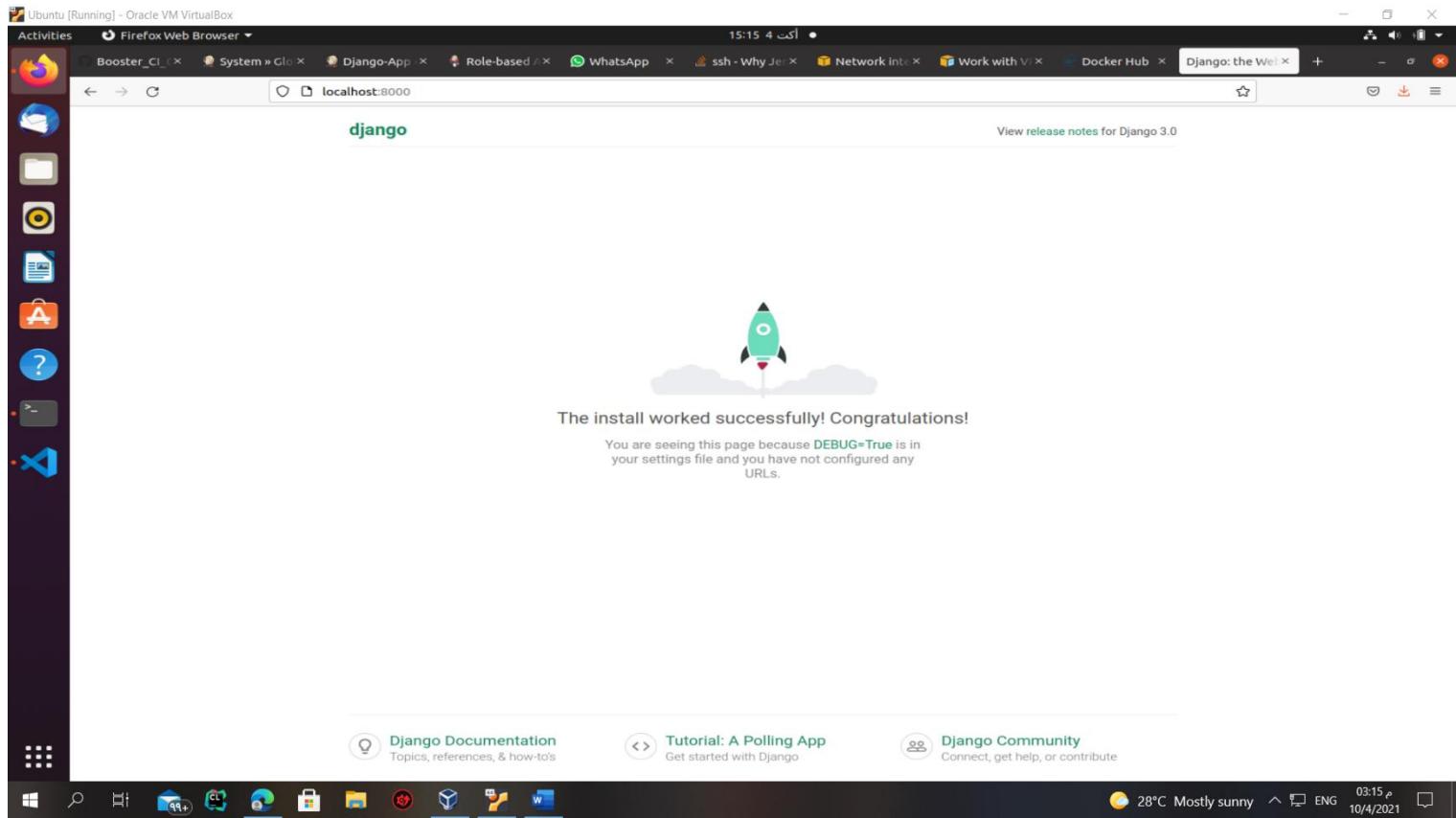
Upgrade to Pro Learn more

Readme

Repository description is empty. Click [here](#) to edit.

28°C Mostly sunny 03:13 10/4/2021

7) Now that the image is uploaded on docker hub, we can access the website through localhost:8000



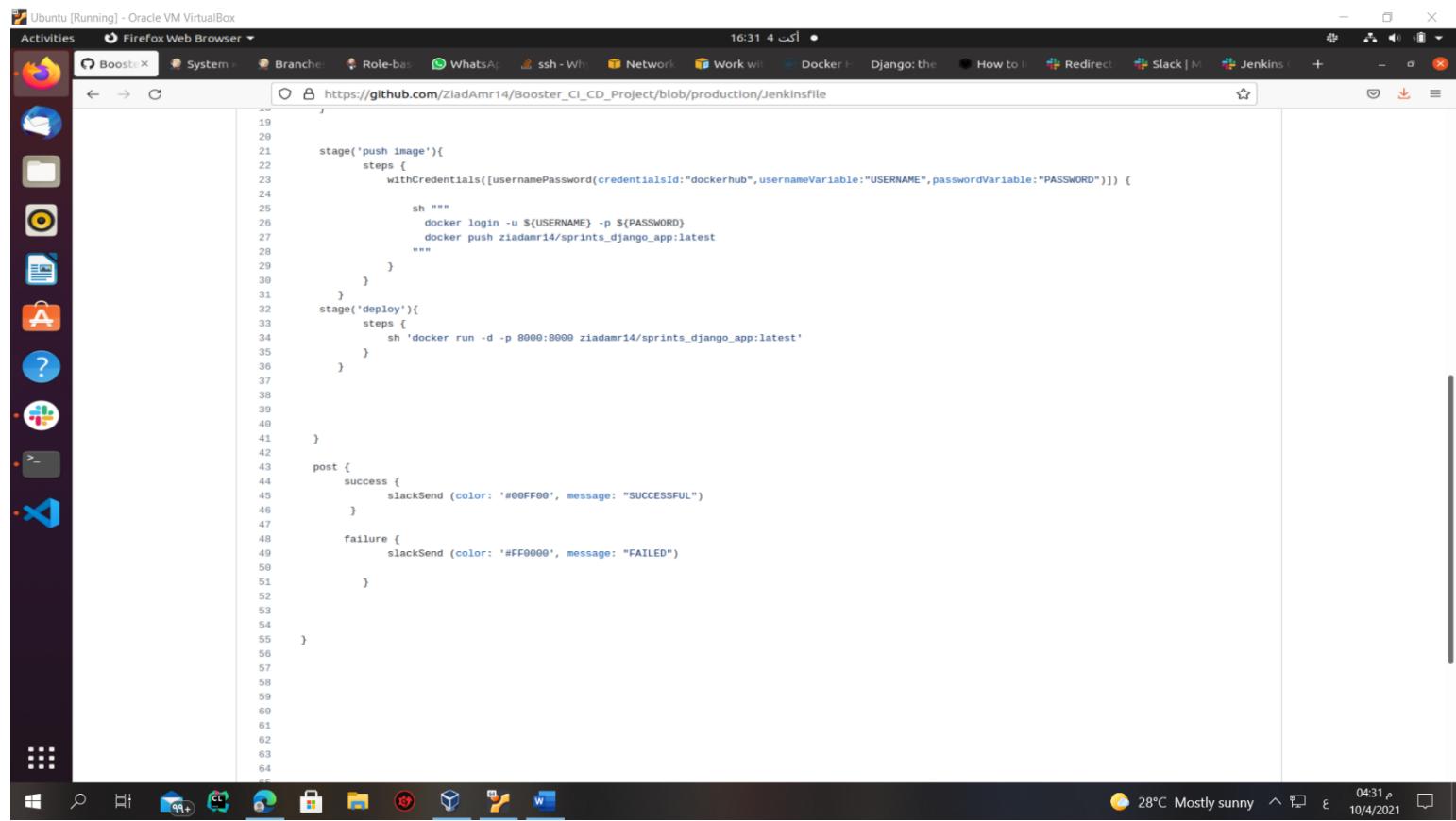
8) Now, what is remaining is to add the slack plugin

a) Add slack plugin

The screenshot shows the Slack configuration interface within a Firefox browser window titled 'localhost:9090/configure'. The interface is divided into sections for 'E-mail Notification' and 'Slack'. Under 'E-mail Notification', there are fields for 'SMTP server' and 'Default user e-mail suffix', both of which are currently empty. There is also a checkbox for 'Test configuration by sending test e-mail'. Under the 'Slack' section, the 'Workspace' is set to 'sprintsglobal'. A 'Credential' dropdown menu is open, showing 'Slack' as the selected option and an 'Add' button. Below it, the 'Default channel / member id' is set to 'djangoapp'. There is another checkbox for 'Custom slack app bot user'. At the bottom of the configuration page are 'Save' and 'Apply' buttons. The top of the browser window shows various tabs and the address bar. The desktop environment includes a dock with icons like Boosterr, System, Config, WhatsApp, ssh - Wh..., Network, Work with Docker, Django: the, How to, Redirect, Slack | M, Jenkins, and others.

The screenshot shows the Slack application interface running on a Windows desktop. The window title is 'Slack | djangoapp | Sprints'. On the left, the sidebar shows the 'Sprints' workspace with a list of channels: '# djangoapp' (selected), '# general', '# random', and '+ Add channels'. It also lists direct messages from 'Ziad Amr' and an option to '+ Add teammates'. The main pane displays the '# djangoapp' channel. A message from 'Ziad Amr' at 3:53 PM states 'joined #djangoapp.'. Another message from 'Ziad Amr' at 4:11 PM says 'added an integration to this channel: jenkins'. A message from 'Jenkins APP' at 4:17 PM says 'Slack/Jenkins plugin: you're all set on <http://localhost:9090/>'. Below the messages, there are two input fields: 'Hello, team!' and 'Let's use this channel for...'. At the bottom, there is a text input field for sending a message to '#djangoapp' with various rich text and media icons. The desktop taskbar at the bottom shows icons for various applications like File Explorer, Task Manager, and a browser. The system tray indicates the date as 10/4/2021, the time as 04:21 PM, and the weather as 28°C Mostly sunny.

b) Now let's add Slack notification in the Jenkins file



The screenshot shows a Linux desktop environment with a dark theme. A Firefox browser window is open, displaying the Jenkinsfile for the 'Booster_CI_CD_Project' repository on GitHub. The URL in the address bar is https://github.com/ZiadAmr14/Booster_CI_CD_Project/blob/production/Jenkinsfile. The Jenkinsfile contains Groovy code for a CI/CD pipeline, including stages for pushing an image to Dockerhub and deploying it, and a post section for sending Slack notifications. The desktop interface includes a dock with various icons and a system tray at the bottom.

```
19
20
21     stage('push image'){
22         steps {
23             withCredentials([usernamePassword(credentialsId:"dockerhub",usernameVariable:"USERNAME",passwordVariable:"PASSWORD")]) {
24
25                 sh """
26                     docker login -u ${USERNAME} -p ${PASSWORD}
27                     docker push ziadamr14/sprints_django_app:latest
28                 """
29             }
30         }
31     }
32     stage('deploy'){
33         steps {
34             sh 'docker run -d -p 8000:8000 ziadamr14/sprints_django_app:latest'
35         }
36     }
37
38
39
40
41 }
42
43 post {
44     success {
45         slackSend (color: '#00FF00', message: "SUCCESSFUL")
46     }
47
48     failure {
49         slackSend (color: '#FF0000', message: "FAILED")
50     }
51
52
53
54
55 }
56
57
58
59
60
61
62
63
64 }
```